PROGRAMMABLE LOGIC APPLICATION NOTES

Richard Katz
Microelectronics and Signal Processing Branch
NASA Goddard Space Flight Center
301-286-9705
rich.katz@gsfc.nasa.gov

This column will be provided each quarter as a source for reliability, radiation results, NASA capabilities, and other information on programmable logic devices and related applications. This quarter will continue a series of notes concentrating on analysis techniques with this issue's section discussing the use of Root-Sum-Square calculations for digital delays. If you have information that you would like to submit or an area you would like discussed or researched, please call or e-mail.

In this edition of *Programmable Logic Application Notes*:

We're Moving ...

There is a new companion www site: http://klabs.org - the old site, http://rk.gsfc.nasa.gov will stay on-line as long as possible. The organization of the www pages will not change much and new material is already being added. Note that url's for the new www site is case sensitive.

2001 MAPLD International Conference

The 4th annual Military and Aerospace Applications of Programmable Devices and Technologies International Conference (MAPLD) will be held September 11-13, 2001, at the Johns Hopkins University/Applied Physics Laboratory. On-line registration is now open.

Some New Topics

While featuring programmable technologies, this year there will be an emphasis on logic, processor, and DSP design. Again, devices, technologies, usage, reliability, fault tolerance, radiation susceptibility, encryption, and applications of programmable devices, processors, and adaptive computing systems in military and aerospace systems are topics for papers. The program will consist of oral and poster technical presentations and industrial exhibits. This conference is open to US and foreign participation and is unclassified. Please see http://klabs.org for additional information.

Papers and Late News Papers

Select papers will be published in the AIAA **Journal of Spacecraft and Rockets**. "Late news" papers, for poster presentation and publication, can still be accepted. Please send in your abstracts and author information to mapld2001@klabs.org.

Program and Invited Speakers

The full technical program will be announced in early July. We have scheduled, as in previous years, a set of interesting invited speakers. Below are the speakers and the titles of their talks; abstracts are available on-line.

Arthur F. Obenshain, Director
Applied Engineering & Technology Directorate
NASA Goddard Space Flight Center

Major General Willie B. Nance, Jr.
United States Army
National Missile Defense Program Executive Officer; System Program Director for the Ballistic Missile Defense Organization
Keynote Address

Dr. David A. Bearden
The Aerospace Corporation
When is A SATELLITE mission too fast and too cheap?

Dr. C. Dianne Martin
Electrical Engineering and Computer Science Department, The George Washington University
Invited Ethics Talk
Recipe for Disaster: Engineering without Ethics

Dr. Roger D. Launius
Chief Historian, NASA

Dr. James E. Tomayko
Carnegie Mellon University
Invited History Talk
From Sequencers to Processors on Early U.S. Spacecraft

Dr. Don Bouldin
University of Tennessee
Platform System-on-Chip Design

Dr. Steve Guccione
Xilinx Corporation
FPGAs for Fault Tolerant Circuits

Seminars

Two classes/seminars will be given at the 2001 MAPLD International Conference on September 10, 2001, before the opening of the Conference. Please be sure to register in advance so there are adequate materials for all participants. These courses will last approximately three hours in length and will be full-length versions of some in-house mini-courses. Abstracts are available on-line.

1. Programmable Logic Devices and Architectures
2. Advanced Design: Designing for Reliability

Failure Reports On-Line

The "Reports Page" on our companion www site, http://www.klabs.org/reports.htm, has been re-organized, with a special section broken out for Failure Reports. These include recent reports such as SMEX/WIRE, NEAR, Cassini probe, and others. Additionally, some general reports such as the *Space Launch Vehicles Broad Area Review Report* and the various reports on the Mars missions are included. Historically significant reports are also being made available for missions such as Skylab and Challenger.

Military Specifications Now On-Line

Military specifications for programmable devices are now being put on-line at http://klabs.org/DEI/References/Military_Specifications.htm. Currently, there are specifications for FPGAs and non-volatile memories. Please let me know if there are any new or additional specifications that you would like to put on the www site. Various indices and cross-references are also made available.

Tutorials and Minicourses

A number of tutorials, some introductory material, and a variety of mini-classes and seminars are now available. Please see http://klabs.org/richcontent/Tutorial/tutorial.htm for additional information.

The tutorial section is broken into two groups:

1. Digital and Programmable Logic
2. Military & Aerospace Systems

The following four mini-classes/seminars have been developed and presented.

1. Programmable Logic in the Space Environment and Advanced Design Techniques
2. Advanced Design: Designing for Reliability
3. Fundamentals of Digital Engineering: Digital Logic
4. Logic Devices and Architectures

Other course planned for development include:

- Fundamental Logic Design: Clocking, Timing Analysis, and Design Verification
- Advanced Design: Performance, Power, and Density In Modern FPGA Architectures
- Advanced Design: Mapping DSP Algorithms to Programmable Device Architectures
- Effective Technical Monitoring
- Advanced Analysis: Computer Performance Modeling for Aerospace Systems
- Fundamental Logic Design: VHDL for High-Reliability Applications - Coding and Synthesis
- Fundamental Logic Design: Verification of HDL-Based Logic Designs for High-Reliability Applications

Loss of Control in "PROMs"

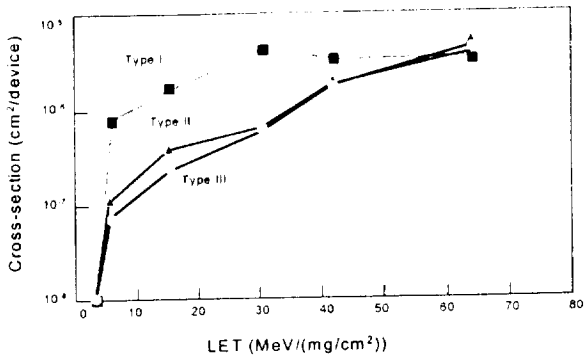
Many non-volatile memory technologies, such as antifuse, fuse, EEPROM and Flash devices are used as PROMs. One must note, however, that there are other sequential circuits on-chip which are not part of the memory array and these must be considered for the reliable operation of the device. Some of these circuits are visible to the user and documented in the device architecture, such as address registers. Others are not visible and include temporary data registers, counters, and other state machines. This section will give a brief overview of what can happen to some of these state machines in commercial devices and some sample data and analysis. The data and analysis is taken from the works of Koga[1] at The Aerospace Corporation and Guertin[2] at the Jet Propulsion Laboratory.

The radiation community defines a *Single Event Functional Interrupt (SEFI)* as a condition where the device stops operating in its normal mode, and usually requires a power reset or other special sequence to resume normal operations. It is a special case of Single Event Upset (SEU) changing an internal control signal. Of course, as electrical engineers, we must use a broader definition of the concept, as a device's state machine may potentially go to any possible state because of ESD, power supply transients, or some other environmental effect. While single event effects testing is effective in identifying failure modes in this class of devices, the conventional term is too narrow. Hence, this section is titled "Loss of Control."

We shall present three case studies. The first two are commercial devices, the Atmel AT28C010 EEPROM and the Xicore X28HC256 CMOS EEPROM. The third case study is the hardened Xilinx XQR1701L serial configuration PROM. The common thread is that each of these devices has failure modes where the power must be cycled to restore functionality.

Atmel AT28C010 EEPROM

The "SEFI" performance of this device is characterized in the graph below. As can be seen, Koga categorizes into three error types which he refers to as Type 1, Type 2, and Type 3.



SEU performance of the Atmel AT28C010 EEPROM, D/C 9706.

Type I errors were manifested by the appearance of repeated errors, once the first error had been detected during ion irradiation. Here, the first error appeared at some point in time, which was tens of reading cycles ("cycle" is defined in Section II) after the exposure had started. Thereafter we observed one error every few cycles. Errors were altered bits in one word at various address locations. Simultaneously with the observation of the first error, the device bias current increased to 26 mA from 20 mA (normal, pre-error condition). The bias current continued to be 26 mA until the reading process stopped. At that time, the current became 0.2 mA (quiescent level). When the device was read again (without power-cycling), the bias current returned to 26 mA and errors appeared again (even without the beam). If the power to the device was shut off and re-started again (power-cycled), the device again functioned properly (i.e., no errors). In one instance we continued the irradiation without power-cycling for a long time, until the device no longer showed any errors. It appeared that the affected bit underwent additional upset, returning to the original polarity and thereby correcting the problem.

The second type (Type II) of errors was manifested by "00" in all address locations, once the first "00" was read at the initial error location. While the reading and tallying of errors continued to take place after the first error, the test computer could not interrogate (read) the device at a fast speed (tallying slowed the reading process), and therefore the bias current became about 10 mA (from 20 mA). This current level remained, even if we stopped reading the device. These errors could be removed only by power-cycling the device.

The third type (Type III) was characterized by occasional errors in a byte, which appeared once in

[1] "Single Event Functional Interrupt (SEFI) Sensitivity in EEPROMs," R. Koga, **1998 MAPLD International Conference**, Greenbelt, MD.

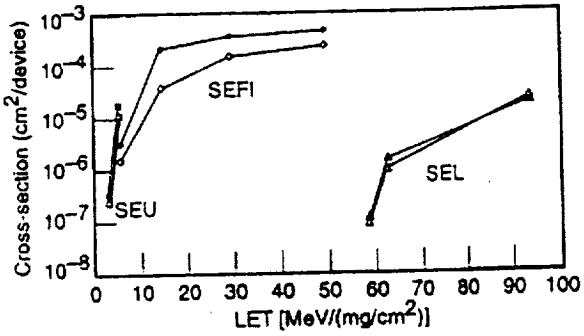
[2] "Single-Event Upset Test Results for the Xilinx R1701L PROM," S. M. Guertin, Jet Propulsion Laboratory, August 24, 2000

many cycles. There was no 'after-effect' for this type of error. In other words, one error appeared independently once in a while.

We consider Types I and II to be SEFIs, while Type III errors were caused by an upset in the output buffer. The test results for the three types of errors are shown in the figure above. Each data point is the averaged values obtained from three test devices. The devices were tested for latchup, in which the bias current was expected to increase drastically. We did not observe any sign of latchup. (The bias current never went up by more than 6 mA as stated above for the Type I errors).

X28HC256 CMOS EEPROM

The X28HC256 device was sensitive to SEU, SEL, and SEFI as shown in the figure below.



SEU Performance of the Xicore X28HC256 CMOS EEPROM, D/C 9140.

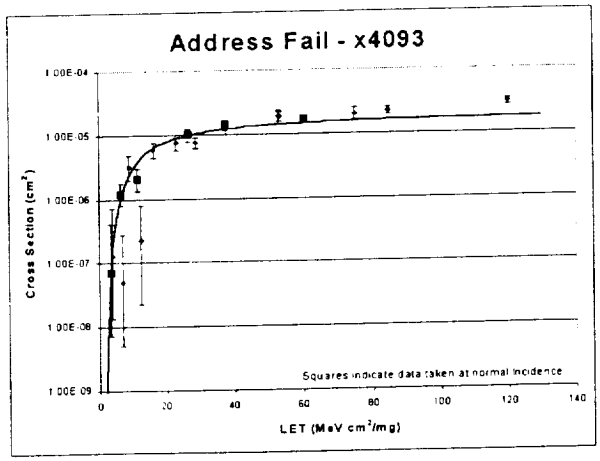
The signature of SEFI was characterized by the condition, in which the device no longer responded to control signals applied to proper control bits, e.g., write enable, device enable, etc. (although the bias current consumption remained the same.) During SEFI, each time the output of an address location was read out, it was almost always FF (hex). This was very similar to Type II in AT28C010 (above), except for the output value. When the bias power was shut off and turned on again, the normal operation resumed. One explanation for this is that the cause of SEFI is the execution of an undefined state vector. For example, a two-bit state vector may be used to express four logic states. If only three have been defined and the fourth has been undetermined, the device may be placed in the undefined fourth logic state. Under normal operating conditions, it is often "impossible" to purposefully attain this condition by design. However, if an impinging ion causes upsets in the state vector, the fourth state may be executed. This may stop the normal operation. Due to a relatively large cross-section for SEFI for this device type, it is not possible to measure the usual SEUs (occurring in the output latches) for LET values beyond 10 MeV/(mg/cm^2).

Xilinx XQR1701 Configuration PROM

According to the manufacturer's data sheet[3], these parts are "Single Event Bit Upset immune." However, a closer look at the radiation characteristics specifies $SEFI_{MAX}$ as 1.2×10^{-5} cm^2/Device as the heavy ion saturation cross section, with 10% of the saturated cross section at an LET of 6.0 MeV-cm^2/mg.

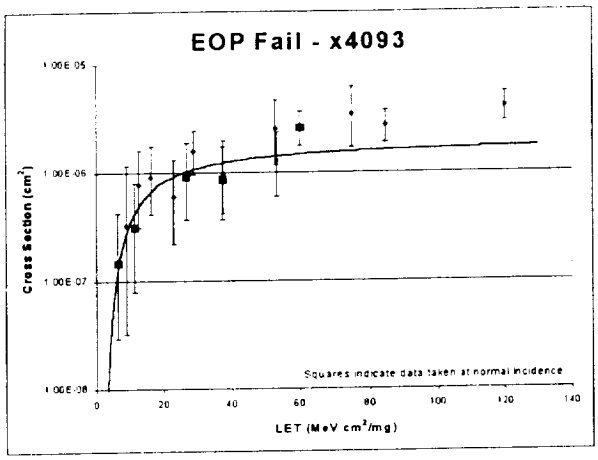
For these tests there were five possible error modes: latchup, bit stream error, address failure, end-of-pass assertion failure, and SEFI. Of these, only the last three odes were observed in the XQR1701L testing. Single-event latchup was not seen with a total fluence of 2×10^7/cm^2 ions at LET = 120 MeV-cm^2/mg. No clearly identifiable bit-stream errors were seen (cross-section less than 5×10^{-6} cm^2/device based on counting statistics).

The first error mode seen in the XQR1701L was *address failure*. Most of the time this appeared to result from a single bit upset in the internal address register (or counter). However, a significant fraction (~40%) of address errors were a reset (to zero) of the address register. Occasionally, two or more bits of the address register were upset; this appears to be consistent with Poisson statistics.

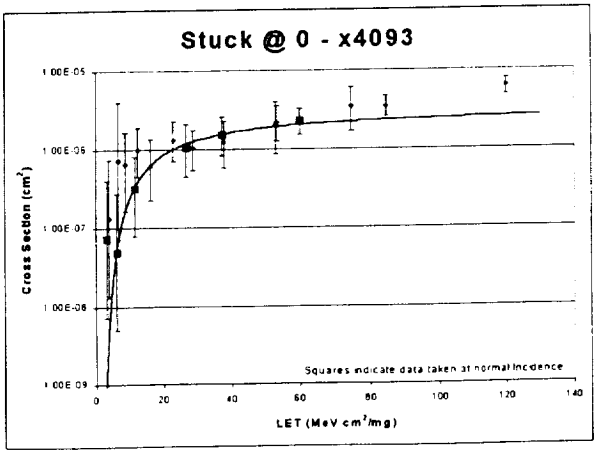


[3] QPRO Series Configuration PROMs (XQ) including Radiation-Hardened Series (XQR), DS062 (v3.0) February 8, 2000

The second error mode observed was the *end-of-part (EOP) assertion failure*. An EOP error is constituted by a discrepancy between the end-of-data (indicated by the device pin) and the actual known end-of-data as verified by address location. Virtually all end-of-part failures were an assertion of the end-of-part pin signal when the data stream was reading out from other parts of the device.



The third and most important error seen on this hardened version of the device was low current functionality interrupt or SEFI, designated *as "stuck at 0"* because the output hangs low. Two features of this mode are (1) the apparent continued operation of the internal address register, and (2) an occasional logical high reading at the output pin. The first feature suggests that this interrupt may only be turning off the output pin. The second seems to confirm this. It seems likely that the output pin is being tri-stated by the error condition, leading to the possibility that occasional high output value may be latched into the testing circuitry.



Analysis Techniques

The following note, *Root-Sum-Square Calculations of Digital Timing Delays*, was contributed by Dr. R. Barto of Spacecraft Digital Electronics. In 1991, Dr. Barto received the NASA Public Service Medal for the Galileo AACS design. This note is the third in a series on analysis techniques.

Root-Sum-Square (RSS) Calculations of Digital Timing Delays

The subject of RSS versus extreme value calculations arises often in worst-case analyses because the calculation of a quantity, e.g., the delay of a digital parts chain, required to be less than some value, will yield a smaller result when calculated by the RSS method than by the extreme value method, making it easier to claim that requirements are met.

The validity of RSS is often debated without exploring its mathematical basis. This report discusses the basis for RSS calculations and the method's limitations. Although the discussion is centered around calculating the propagation delays of digital circuits, the basic theory and conclusions apply to any application of RSS.

What is the Difference Between an RSS and an Extreme Value Calculation?

Let a timing chain contain gates $G_1,...,G_n$, having data book typical delays $T_1,...,T_n$ and maximum delays $M_1,...M_n$. The goal is to determine whether the delay D of their series combination is less than some value. The extreme value for the delay would be given by

$$D_{MAX} = \sum_{i=1}^{n} M_i$$

The delay for the same chain calculated by the RSS method is given by

$$D_{RSS} = \sum_{i=1}^{n} T_i + \sqrt{\sum_{i=1}^{n} (M_i - T_i)^2}.$$

Clearly, D_{RSS} will be less than D_{MAX}, and the two will be equal only if the individual typical and maximum values are equal.

How is the RSS Formula Derived?

The RSS formula is derived by analogy from the statistical calculation that would be used to estimate the variance of the delay chain if the means and variances of the individual delays were known.

Assume that $X_1,..., X_n$ are independent random variables with means $\mu_1, ..., \mu_n$ and variances $\sigma_1^2, ..., \sigma_n^2$. Then it can be shown that the random variable Y defined by

$$Y = \sum_{i=1}^{n} X_i$$

has mean μ_Y and variance σ_Y^2 value given by

$$\mu_Y = \sum_{i=1}^{n} \mu_i, \quad \sigma_Y^2 = \sum_{i=1}^{n} \sigma_i^2.$$

If the data book specifications for the G_i were given as $3\sigma_i$ values m_i and means μ_i, then their standard deviations could be found by

$$\sigma_i = \frac{m_i - \mu_i}{3}.$$

Then, the 3σ value for the length of the timing chain would be given by

$$D_{3\sigma} = \mu_Y + 3\sigma_Y = \sum_{i=1}^{n} \mu_i + \sqrt{\sum_{i=1}^{n} (m_i - \mu_i)^2}.$$

The correlation between the forms of D_{RSS} and $D_{3\sigma}$ equations is obvious, but the two equations are equivalent only if:

1. The typical values T_i are the same as the means μ_i;
2. The maximum values M_i are the same as the 3s values m_i.

These assumptions are not guaranteed to be valid. Data books do not give statistical definitions of typical and maximum values, although it is assumed there will be no gates with propagation delays longer than M_i. Thus, the transformation by which the D_{3s} equation becomes the D_{RSS} equation by substitution of typical and maximum for mean and variance is erroneous.

What Effects Do Parameter Distributions Have on the Validity of RSS Calculations?

If the X_i had normal distributions, then it can be shown that Y also has a normal distribution. If the G_i had normal delay distributions X_i, then their series delay D would be normally distributed and have only a 0.0013 probability of being longer than $\mu+3\sigma$, assuming, of course, that μ and σ are known. Data books do not give part parameter distributions, and there are reasons why a normal distribution might not be expected:

1. Experience shows that space qualified parts from a given lot are closely matched in performance rather than exhibiting a normal distribution - a result of the intense scrutiny to which they are subjected. As parts from different lots are mixed together, the resulting distribution could become multimodal or flat. The mean values of the final parameter populations could differ greatly from data book typical values.
2. Some digital parts are speed graded by selecting the fastest parts of an initial population for sale at a higher price. If the initial delay distribution was normal, then the distributions of the fastest and slowest sub-populations will not be.

Thus a normal parts distribution should not be assumed unless it can be proven.

If the X_i are not normally distributed, then it is still possible to make a statistical statement about Y using Chebyshev's Theorem:

$$P(\mu - k\sigma < Y < \mu + k\sigma) \geq 1 - \frac{1}{k^2}.$$

Thus the probability that the timing chain delay would be within 3σ of μ is at least 0.8888. Assuming a symmetrical distribution about μ, there is a 0.0555 probability that the delay would be longer than $\mu+3\sigma$. This is much larger than the 0.0013 probability given by the normal distribution case.

It might be argued that even if the X_i are not normal, the Central Limit Theorem will guarantee that Y will be almost normal. This ignores the fact that timing chains are generally very short. It is uncommon to have a delay chain as long as 10 gates, and 2 to 5 gates is a normal length. The Central Limit Theorem argument could not be considered valid if the length was less than about 30 gates.

Why Cannot Systems Be Tested for the Timing Problems that RSS Doesn't Find?

In a commercial environment, it might be argued that using RSS is acceptable because the systems with excessive delays will fail in test and not be shipped. The parts are assumed to not appreciably degrade, implying that system margins will not decrease significantly: most commercial systems have short lifetimes, relative to some space missions, and are not subjected to temperature extremes or radiation. This argument is specious because it assumes that the tests will be sufficiently exhaustive to exercise and monitor all operational modes and conditions, which is highly unlikely. For example, a flip-flop set-up time not met might manifest itself as a decrease in performance that would be noticed only under high system load conditions. The possibility of more serious erroneous operation resulting from flip-flop metastability would be hidden.

Testing for timing problems is clearly not an option in the spacecraft environment. In analyzing spacecraft circuitry, the parts parameters must be degraded for age, radiation, and temperatures that most commercial systems will not see. making timing margins at beginning of life much larger than the calculated end-of-life margins. Thus a circuit can have an end-of-life timing delay longer than its required maximum value, pass all pre-launch tests, yet fail later in the mission because of insufficient margin. There is no method to test for excessive timing de lays other than measuring every one. Finding timing problems this way is expensive in both cost and schedule, even if no redesign is required as a result of the measurements.

Conclusion: Use RSS with Caution

Given the above, two points should be clear concerning the use of RSS:

1. RSS calculations should use the means and variances of the parts that will be used, not data book typical and maximum values, and should not without proof assume normal parameter distributions. This almost always requires a lengthy and expensive part measurement program.
2. RSS is a statistical technique and does not guarantee that timing delays will be within their required values. There is a non-zero probability that a system passing an RSS-based timing analysis will have timing problems.

RSS analyses should be viewed with suspicion. Extreme value computation is the only method that guarantees that when the analysis says there are no timing problems, there *really are no timing problems!*

Programmable Logic Device Failures

While the reliability of programmable logic devices is quite good, they do have a measurable failure rate. A database of device failures is being established. This will enable us to measure field reliability, categorize the failure modes, and spot trends as early as possible.

Failures will be divided into three classes:

1. Hardware
2. System software
3. Design error

The first group, hardware failure, will obviously include devices that fail as a result of a manufacturing defect, as an example. Failures because of system application, such as an over voltage transient or ESD, should also be included. Programming failures for antifuse-based devices are not counted as failures, if the error is detected by the programmer's diagnostics. Programming yield information is also desired.

The second group of failures will include front-end software such as logic synthesizers, optimizers and translators, and back-end software such as place and route tools, simulation models, or timing analyzers.

The last group, design error, should include things such as logic design error, specification error, application error, assembly error, etc.

We are asking aerospace engineers to send in information about failures that they observe; all submissions will be kept anonymous, if desired. You may e-mail your information to rich.katz@gsfc.nasa.gov. Please include the following information, at minimum:

- Part Model Number
- Part Date and Lot Code
- Contact Name and Information
- Type of Failure
- Reason for Failure
- Supporting information, as available

RT54SXS Series Devices
SX32S and SX72S Pin Incompatibilities

It has been noted that the SX32S and the SX72S are not 100% pin compatible. That is, a design implemented in a SX32S device can not grow into a SX72S advice unless certain pin restrictions are observed. The incompatibilities have arisen from the increased gate count of the larger device and the need for additional power and ground pins. This section will discuss the differences between these two devices.

In addition to the number of available pins per device-package combination, in migrating from an RT54SX32S to an RT54SX72S there are 3 cases:

1. I/O → Power Connection {GND, VCCI, VCCA}
2. I/O → {QCLK, I/O}
3. NC → I/O

Considerations for the V_{CCR} supply will also be discussed.

Available Pins Per Device and Package

Device	User I/Os (including clock buffers)	
	CQFP208	CQFP256
RT54SX32S	173	227
RT54SX72S	170	212
Loss	3	15

CQFP208: RT54SX32S vs. RT54SX72S

Package: CQFP208		
Pin Number	RT54SX32S Function	RT54SX72S Function
18	I/O	GND
19	I/O	V_{CCA}
25	NC	I/O
65	NC	I/O
74	I/O	{QCLKA, I/O}
83	I/O	V_{CCI}
84	I/O	{QCLKB, I/O}
116	I/O	GND
117	I/O	V_{CCA}
132	NC	I/O
178	I/O	{QCLKD, I/O}
187	I/O	V_{CCI}
190	I/O	{QCLKC, I/O}

CQFP256: RT54SX32S vs. RT54SX72S

Package: CQFP256		
Pin Number	RT54SX32S Function	RT54SX72S Function
17	I/O	V_{CCI}
36	I/O	V_{CCA}
37	I/O	GND
47	I/O	V_{CCI}
56	I/O	GND
73	I/O	V_{CCI}
89	I/O	{QCLKA, I/O}
98	I/O	{QCLKB, I/O}
120	I/O	V_{CCI}
142	I/O	V_{CCI}
143	I/O	GND
144	I/O	V_{CCA}
162	I/O	V_{CCA}
183	I/O	V_{CCI}
202	I/O	V_{CCI}
218	I/O	{QCLKD, I/O}
228	I/O	V_{CCA}
231	I/O	{QCLKC, I/O}
249	I/O	V_{CCI}

RT54SXS Incompatibility with V_{CCR} of the RT54SX Series

Note: In moving from an RT54SX to an RT54SXS, the V_{CCR} supply is not used in the RT54SXS series. On the RT54SX32S, it is a NC. For the RT54SX72S in the CQFP208 package, **there are two incompatibilities.**

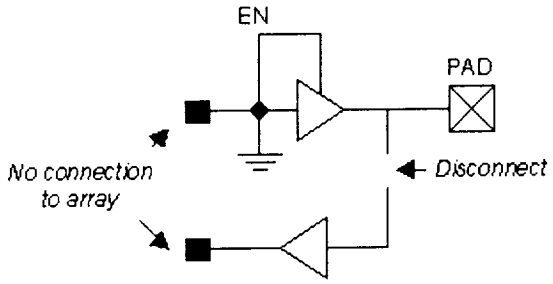
V_{CCR} Usage in the RT54SX Series and Mapping onto Functions in the RT54SXS Series Devices			
	RT54SX V_{CCR}	SX32S	SX72S
CQFP208	25	NC	**I/O**
	80	NC	NC
	132	NC	**I/O**
	182	NC	NC
CQFP256	159	NC	NC
	223	NC	NC

Termination of Unused I/O's in the RT54SX and RT54SXS-Series Devices

User I/O

For RT54SX and RT54SX-A series devices, unused I/Os [this does not include CLK and HCLK in all models] may be left unconnected on the board design. The output buffer for each I/O module is disabled; this is different than Act 1 and Act 2 devices where they were programmed as active and driving low. In the case of the RT54SX and RT54SXS, the input stage is disabled. The buffer is physically implemented as a NAND function and a fuse disables it, such that there is no totem-pole current.

The Actel guru system has a drawing of the I/O module and it's configuration when not used. This is electrically incorrect and is shown below:



This drawing implies that the input buffer is floating and can not be terminated. Obviously, this configuration would lead to totem-pole current if the input would move from either of the rails [which can happen for a number of different reasons]. Furthermore, it suggests that the design engineer needs to configure each unused I/O to prevent this condition. This should not be necessary.

Termination of CLK, HCLK, and QCLK Inputs

The **HCLK** input on all models, if not used, must be terminated by the user on the board. There is no output buffer available on chip to terminate the input.

For the **CLKA/B** inputs, for the **RT54SX16**, **RT54SX32**, and the **RT54SX32S** [all so far except the RT54SX72S], the inputs, if not used, must be terminated by the user on the board. There is no output buffer available on chip to terminate the input.

For the **CLKA/B** inputs on the **RT54SX72S**, the CLKA/B pins can be configured as user I/O. I have a letter into Actel to find out if the software will automatically terminate these particular inputs if unused. I tried to terminate the A54SX72S CLK inputs with an OUTBUF and it did not work with the R1-2000 software.

For the **QCLKA/B/C/D** inputs in the RT54SX72S, the inputs, if not used as quadrant clocks, can be configured as user I/O. Current software will not terminate these outputs for you and the user must do so on the circuit board. A pull-down resistor is recommended as "normal" Actel software terminates unused pins to ground through an OUTBUF driving low, and this will help keep designs compatible with any future software changes. This information is current as of the latest data sheet available, version "Advanced v0.2."

Unused Clock Needs Termination On Board?

	Part Number	HCLK	CLK	QCLK
RT54SX	RT54SX16	Yes	Yes	N/A
	RT54SX32	Yes	Yes	N/A
RT54SXS	RT54SX32S	Yes	Yes	N/A
	RT54SX72S	Yes	Yes*	Yes*

* A pull-down resistor or enabled OUTBUF.

Can Unused Clock Be Used as User I/O?

	Part Number	HCLK	CLK	QCLK
RT54SX	RT54SX16	No	No	N/A
	RT54SX32	No	No	N/A
RT54SXS	RT54SX32S	No	No	N/A
	RT54SX72S	No	Yes	Yes

RT54SXS Series Devices I/O Determinism During Power Transitions

It is well known[4,5,6] that many programmable devices' outputs are not well controlled during the power transition. Effects include inputs sourcing or sinking current, devices not following their truth tables, output "glitches," or buffers either active or in tri-state unexpectedly. While there are differences in design and behavior for Act 1, Act 2, Act 3, and RT54SX series devices, they all, from a design and analysis standpoint, must be considered out of control during the power transition. If the characteristics of these devices are not taken into proper consideration during the design stage, loss of mission may occur; as qualification by test is inadequate.[7]

[4] "A Power-On Reset (POR) Circuit for Actel Devices," Actel Corporation, September 1997.
[5] "Startup Design and Analysis Note," R. Katz.
[6] "Programmable Logic Application Notes," R. Katz, **NASA EEE Links**, March, 1997.
[7] "Small Explorer WIRE Failure Investigation Report," R. Katz, May 27, 1999.

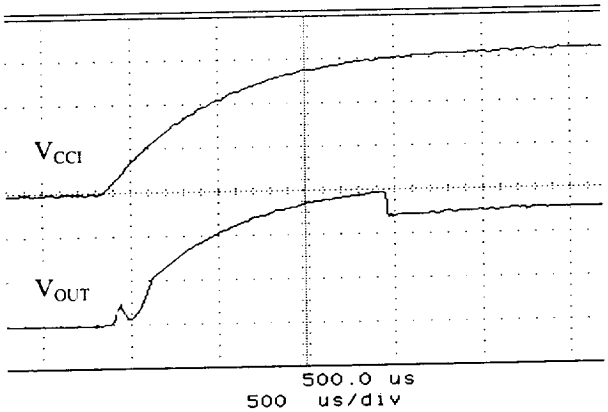
The design of the RT54SXS series of devices was intended to supply deterministic, safe, behavior of the I/O's during power transitions.[8] The features in the design include:

- Individually selectable, optional, pull-up or pull-down resistors during power-up
- Output buffer tri-state during the power transition
- Release of resistors and output buffer 50 ns after the device is operating to allow signals to propagate through the device and stabilize

The properties for each pin are selected in the Designer software in the Pin Edit function.

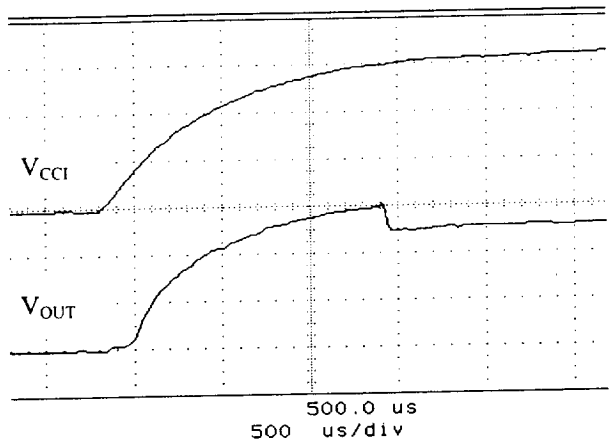
The initial evaluation performed to date has shown some problems with these features. Additionally, two other users have written in with similar type problems. The problems are being worked with Actel and follow-up information will be presented in the next *Programmable Logic Application Notes* column. At this time, it would be prudent to design assuming that the device's I/O's are not controlled during power transitions.

Three examples from our preliminary evaluation are included below. These are not exhaustive of all cases and should not be used to characterize the device's behavior. This is simply a first, preliminary look. The sample used for this evaluation was an RT54SX32X, S/N LAN4801 packaged in a CQ208, 4Strings pattern, pre-irradiation. All voltages are scaled at 1V/Division.

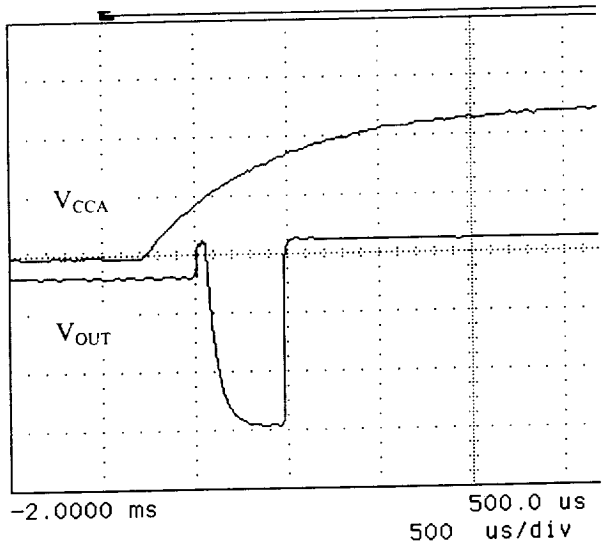


An RT54SX32S output during power-on transition. The top trace is V_{CCI} and the bottom trace is V_{OUT}.

The OUTBUF module on pin, IODIO(7) was configured with PCI levels, high-slew, no pull-up or pull-down resistors, and hot-swap on. Aside from the initial glitch, we see that V_{OUT} tracks V_{CCI}; it was expected that the output would be in tri-state until the device was operational.



Another RT54SX32S output during power-on transition. The top trace is V_{CCI} and the bottom trace is V_{OUT}. The OUTBUF module on pin, O_AND4 was configured with PCI levels, low-slew, a pull-down resistor, and hot-swap on. Again, we see V_{OUT} tracking V_{CCI}; it was expected that the output would be held low by the pull-down resistor until the device was operational.



The last case shown here reverses the order of power application, with V_{CCI} stable at 3.3V before V_{CCA} is applied, for the same output above. Here we see, after a small glitch, the pull-down resistor enabled. However, for a safe power-on sequence, the user would expect the output to be stable at 0V and not propagate a logic high to external circuits.

[8] "Radiation-Hardened/High Reliability Programmable Logic Using Modified Commercial-off-the-Shelf RTSX32S," B. Cronquist, R. Katz, J.J. Wang, J. McCollum, I. Kleyner, I. Brill, W. Parker, and K. A. LaBel, **2000 MAPLD International Conference**, September, 2000.

RT54SXS Series Devices
Input Thresholds

The RT54SX32S and RT54SX72S have logic input thresholds that are selectable on an individual I/O basis (except for the HCLK, RCLK, and DCLK clock buffers, which are globally set by programming the SSIG fuse). The levels are:

1. LVTTL
2. 3.3V PCI
3. 5V CMOS
4. 5V PCI/TTL

All inputs are 5 volt tolerant. Note that the 5V CMOS input threshold, designed for increased noise immunity, is not available in the A54SX-A series, which are desirable for inexpensive prototyping.

Engineering evaluations of these inputs are now underway and the results of these evaluations will be included in this application note. To date, the evaluations have focused on the new input level, 5V CMOS.

Measurements on a production RT54SX32S have shown a 5V CMOS logic threshold of approximately 2.8 V. Test conditions were T = 25°C, V_{CCI} = 5.0 VDC, and V_{CCA} = 2.5 VDC. This was performed on an INBUF hard I/O macro.

Attempts to measure 5V CMOS logic thresholds on the clock buffers (CLKBUF, HCLKBUF) failed. Following up with the manufacturer, it has been learned that the devices, because of the "hot-swap" I/O capability, can not have the 5V CMOS thresholds on the two different types of clock buffers. For the HCLK, which is hardwired, there are no "work-a-rounds." The section below will show a work-a-round for the CLKBUF interface and a sample timing measurement for analysis.

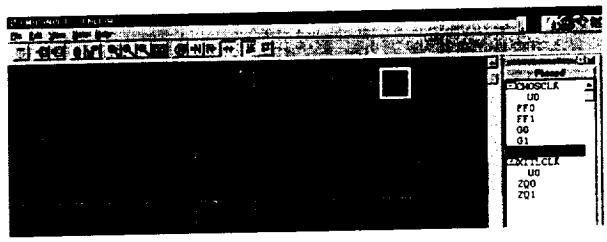
The global clock network can be accessed from either a device pin (using CLKBUF) or any internal signal (using CLKINT). Exploiting this feature, CMOS clock thresholds for the two global clocks can be achieved by bringing the clock signal into the chip via an INBUF macro and then routing it to a CLKINT macro.

A small test chip was designed to evaluate the impact of this work-a-round and analyzed with TIMER and Designer R1-2000. The evaluation circuit is shown below.

Circuit used for examining the timing delays when inserting an INBUF to construct clock buffers with 5V CMOS compatible logic thresholds.

The design was imported into Designer R1-2000 and then placed and routed. Two runs were made. The first run was hand-placed before routing; the second used fully automatic place and routing. No major differences in timing were observed.

The automatic placement software placed the INBUF macro close to the CLKINT macro that is critical for good performance.



Automatic placement of INBUF macro. The INBUF macro is configured with 5V CMOS compatible logic thresholds.

Static Timing Analysis Results

Worst-Case Conditions
Temperature = 125 °C
Array Voltage = 2.3 VDC
Speed Grade = STD

TTL CLKBUF Buffer

Macro	Delay	Total
OUTBUF	4.40 (r)	13.00
DF1	1.70 (r)	8.60
CLKBUF	6.9 (r)	6.90

CMOS INBUF + CLKINT Macros

Macro	Delay	Total
OUTBUF	4.40 (r)	12.70
DF1	2.10 (r)	8.30
CLKINT	4.60 (r)	6.20
INBUF	1.6 (r)	1.60

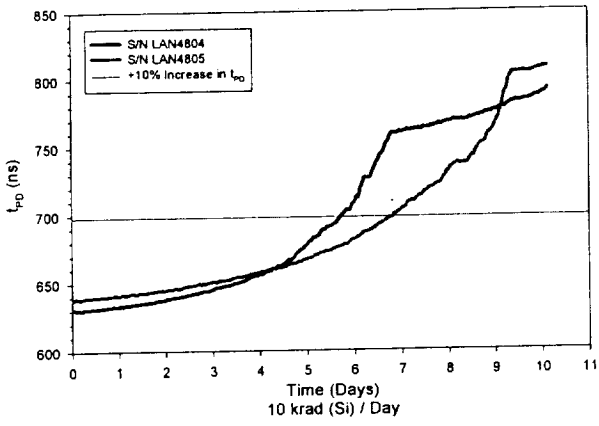
SX-A and SX-S Series Devices Power Sequencing Revisited

In the last edition of *Programmable Logic Application Notes*, it was noted that there were no requirements or limitations on power sequencing for RT54SX-S series devices. However, laboratory experiments on RT54SX32S devices have shown some anomalous behavior.

For the RT54SX32S devices that were tested, if V_{CCI} is applied before V_{CCA}, then there can be some DC current draw on the V_{CCI} supply pins. The amount of current was part dependent and ranged from approximately 1 mA to over 10 mA. Applying V_{CCA} would cause the current into the V_{CCI} supply pins to return to expected values. The amount of current was also a function of V_{CCI}, with higher voltages on V_{CCI} resulting in higher currents. The laboratory tests included voltages in the range of $2.25\ V < V_{CCI} < 5.5\ V$.

RT54SXS Series Devices Propagation Delay and Radiation

It is noted that the RT54SX32S FPGAs that have been tested for total dose have shown that the propagation delay (t_{PD}) is sensitive to radiation. The figure below shows representative results from a lot qualification test. A reference line was added 10% of the pre-irradiation tPD value, as a ± 10% variation on propagation delay is often used in worse-case analyses.



RT54SX32S device t_{PD} radiation sensitivity. Test results for two devices, packaged in a CQ208, from wafer lot T25JSP03A.

Variables for Designer Software Placement and Routing

The following list contains some useful variables for the placement and routing of Actel devices.

LAYOUT_MODE determines the type of layout that will occur. Select STANDARD for typical layout or select TIMING_DRIVEN for Direct Time Layout.

PLAINC determines whether the current placement will be based on the previous placement. Select 0 when you desire the current placement to be based on the previous placement or select 1 when you want the placement to start fresh.

RESTRICTPROBEPINS determines whether the SDI, DCLK, PRA and PRB0 pins will be used as I/Os. Select 0 to prevent the usage of the special probe pins or select 1 to use the probe pins.

RESTRICTJTAGPINS determines whether the TDI, TDO, TMS and TCLK pins will be used as I/Os. Select 0 to prevent the usage of the special JTAG pins or select 1 to use the JTAG pins. Yes, it is an issue and a call has been placed about the TRST* pin.

PLACEOVERNIGHT determines the amount of time the placement algorithm will run. Select 1 to do placement that will run 5 to 6 times longer than normal or select 0 for normal run time. Note some sample runs on a simple circuit have shown that better timing performance is sometimes obtained for normal run time.

PLACESEEDRANDOM causes the placement algorithm to start from a different starting point. Every starting point produces a slightly different result. Set this to a positive integer x, such that $1 < x < 2^{31} - 1$.

LAYOUTINFO causes layout to record layout information in a file whose name is the value of LAYOUTINFO.

LAYOUTAFL causes layout to output an AFL information in a file whose name is the value of LAYOUTAFL.

Virtex FPGA: SEU Performance

The following summarizes Virtex SEU device performance. Much of this material is extracted from the Fuller[9] paper at the 2000 MAPLD International Conference. The paper also discusses mitigation strategies. Heavy ion SEU performance, proton susceptibility, and an estimate of on-orbit performance will be discussed.

Summary of Latch Types

A breakdown of the bits in the XQVR300 is shown in the table below. Similar to an earlier analysis[10] of SRAM-based FPGAs, we can see that the ratio of storage increases by approximately one order of magnitude from the user flip-flops, to user SRAM, to configuration bits. That is, in the case of the XQVR300, to a first order approximation, there are 10^4 user bits, 10^5 SRAM bits, and 10^6 configuration bits. Based on bit count and SEU performance in the charts below, it is clear that any SEU mitigation strategy will have to ensure that the configuration remains valid and that errors are not propagated.

Note that the referenced paper discusses that not all routing bits will affect a particular circuits function; that is, an upset can be a "don't care." This is similar to what was observed with antifuse damage, with many broken antifuses not resulting in functional failure for a particular test and evaluation circuit.

Latch Types in the Virtex XQVR300 FPGA

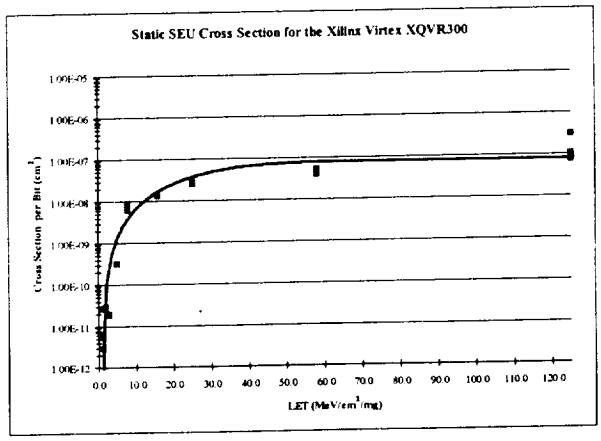
Latch Type	Number of Bits	Approx. No. Bits
CLB	6,144	6×10^3
IOB	948	1×10^4
LUT	98,304	1×10^5
BRAM	65,536	7×10^4
Routing and Other Bits	1,579,860	2×10^6

Loss of Functionality

Additionally, a few bits affect the device configuration. Note that these devices also do not include the optional TRST* pin in the IEEE JTAG 1149.1 interface, which holds the Test Access Port (TAP) controller in the `TEST-LOGIC-RESET` state.[11] Loss of functionality in this device was reported for both heavy ion tests and proton tests.

Heavy Ion SEU Performance

The threshold LET (LET_{TH}) for this device is reported as 1.2 MeV-cm^2/mg. During testing, contention[12] was observed internal to the part, indicated by small fluctuations in the power supply current; no catastrophic upsets were observed. Fuller, et. al., notes that the internal drive capability in the Virtex family is limited and that there is no long-term reliability hazard. Loss of functionality of the part was observed with a complete loss of configuration. The LET_{TH} for this effect was between 8 and 16 MeV-cm^2/mg; low enough to be a concern in the proton environment. It took a fluence of approximately 10^5 ions/cm^2 for functional loss.



Static heavy ion bit upset cross-section vs. LET for the Virtex FPGA.

[9] "Radiation Testing Update, SEU Mitigation, and Availability Analysis of the *Virtex* FPGA for Space Reconfigurable Computing," Fuller, et. al., **2000 MAPLD International Conference**, Laurel, MD.

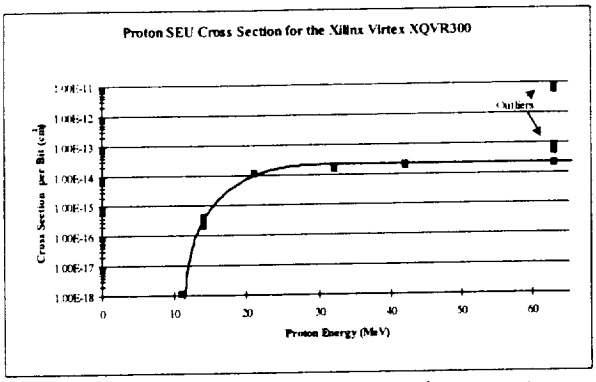
[10] "Current Radiation Issues for Programmable Elements and Devices," R. Katz, et. al., **IEEE Trans. on Nuclear Science**, December 1998.

[11] "Current Radiation Issues for Programmable Elements and Devices," R. Katz, et. al., **IEEE Trans. on Nuclear Science**, December 1998.

[12] This was predicted and analyzed in "SRAM based Re-programmable FPGA for Space Applications," J. Wang, R. Katz, J. Sun, B. Cronquist, J. McCollum, T. Speers, W. Plants, *IEEE Transactions on Nuclear Science*, Vol 46, No. 6, pp. 1728 – 1735, December 1999.

Proton SEU Performance

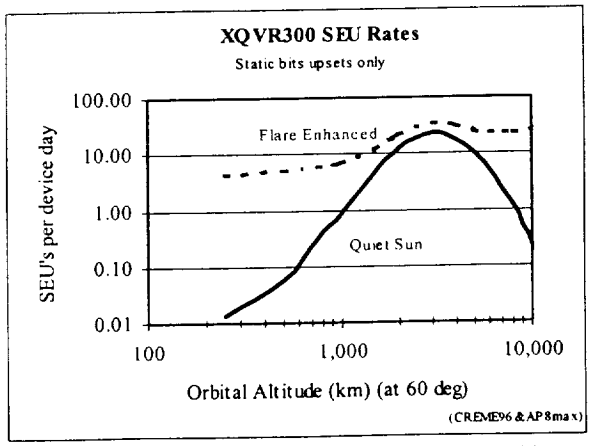
The figure below summarizes the results of the Virtex proton testing. Note the identification of outliers in the chart indicating that the configuration control circuit upset mode was observed.



Static proton induced bit-upset cross-section vs. proton energy for the Virtex FPGA.

On-Orbit SEU Rate Estimation

Every orbital scenario is different, however, it is useful to calculate an upset rate for sample orbits. For the sake of example, several circular orbits were modeled assuming a 60-degree inclination angle and operation during solar maximum. Using these assumptions and the CREME96 model, the upset rate for the device can be estimated. With the knowledge of both the proton and heavy ion response, this modeling tool provides both an orbital average upset rate as well as the higher rate that would occur during periodic solar flare events. The graphs below indicate some of the results of the modeling effort.



XQVR300 upset rate estimate. This plot is obtained by calculating the hypothetical sensitive volume of the XQVR300 as the product of all of the bits in the FPGA and the average cross-section.

HDL Design Techniques

The following note, *Minimizing HDL Design Errors*, was contributed by Ben Cohen of VhdlCohen Publishing (http://www.vhdlcohen.com/).

MINIMIZING HDL DESIGN ERRORS

Abstract

This paper discusses processes, methodologies, and classes of tools necessary to minimize ASIC and FPGA design errors.

INTRODUCTION

To understand the processes necessary to minimize errors it is important to understand the definition of an "error", and where errors can be introduced, along with the cost consequences of those errors. The dictionary[i] defines "error" as the following:

1. An act, an assertion, or a belief that unintentionally deviates from what is correct, right, or true.
2. The condition of having incorrect or false knowledge.
3. The act or an instance of deviating from an accepted code of behavior.
4. A mistake.
5. Mathematics: The difference between a computed or measured value and a true or theoretically correct value.

Other definitions for errors include the improper design performance in terms of speed / area / power / electrical interfaces / radiation / fault recovery. The dictionary further extends the concept of error conditions by providing two examples:

1. **Division By Zero**: An error condition caused by an attempt to divide a number by zero, which is mathematically undefined, or by a number that is sufficiently near to zero that the result is too large to be expressed by the machine. Computers do not allow division by zero, and software must provide some means of protecting the user from program failure on such attempts.[ii]
2. **General Protection Fault** (GPF): The error condition that occurs in an 80386 or higher processor running in protected mode (such as Windows 3.1) when an application attempts to access memory outside of its

authorized memory space or an invalid instruction is issued.[2]

These definitions point to errors introduced during the various stages of the design process. Those errors attributed to *incorrect or false knowledge* or to *a belief that unintentionally deviates from what is correct, right, or true* are generally initiated because of poorly documented requirements and a misunderstanding of what the design must accomplished, and with what performance. Other errors attributed to *deviation from an accepted code of behavior* are generally introduced because of poor use of coding standards and style that makes code very difficult to communicate, read, and debug. Simple *Mistakes*, such as syntax errors, are often introduced because of poor use of tools, such as editors or linting tools, which verify the sanity of the code.

Other errors conditions can be introduced at runtime, such as divide by zero, general protection fault, or taking severe erroneous actions (e.g., blowout) of a mission critical subsystem. The anticipation and handling of those runtime errors must be addressed early in the design process.

DESIGN PROCESS AND ERROR CONSTRAINING

The typical front-end phases of a design include definition of requirements, architectural design, verification plan, behavioral and/or RTL design, synthesis, timing analysis, design applications, documentation and delivery, as highlighted in Figure 1.

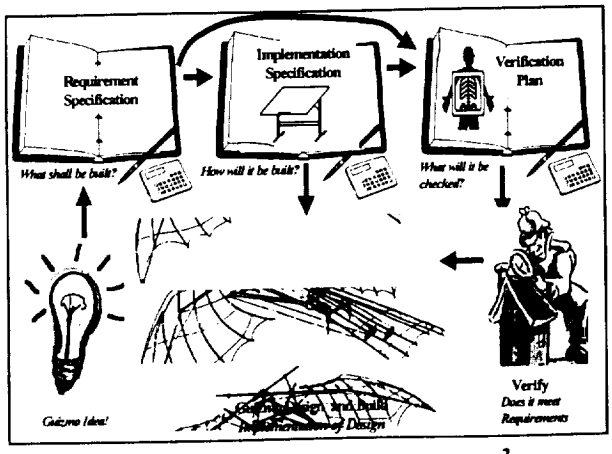


Figure 1. Typical Design Processes [3]

REQUIREMENT SPECIFICATION

Poorly specified requirements will introduce expensive errors because the requirements drive the design. Thus, if there were only one phase where the minimization of error emphasis should be emphasized, it is the definition of the requirement specification[iii]. An error in requirements will lead to a design that meets incorrect requirements, and thus is in error (i.e., *Garbage In → Garbage Out*). To correct the problem means either a complete or partial redesign, or a change in requirements to adapt around the error (i.e., change the error into a design feature, or make lemonade out of a lemon).

ARCHITECTURAL SPECIFICATION

The implementation specification defines the architecture, intellectual properties (IPs), and design tools. An improper architectural plan, use of IPs, and tools will lead to design errors because the design may not meet the goals, performance, or intended costs. The architectural specification drives the hardware design cycle. It is more economical to make changes at this level of the design process rather than later in the design cycle.

VERIFICATION PLAN

The verification plan *is a specification for the verification effort. It is used to define what is first-time success, how a design is verified, and which testbenches are written*[iv]. The process of writing a verification plan causes the design and requirement team to get a deeper understanding of the performance and holes in the specifications. Requirement issues often surface during the definition of the verification plan because it forces the revisit of the requirements and the challenges of verifying correctness of the design. Poorly stated or ambiguous requirements become obvious in this exercise.

DESIGN IMPLEMENTATION AND MANAGEMENT

This is where the "art" of circuit design, "reuse", knowledge of the "design language" (e.g., HDL) and level of applicability, agility in tool usage all come into play, along with an understanding of the requirements, and a vision of how lower level architectural implementations impact performance.

Art is a non-tangible attribute. Thus, to minimize design errors, it is important to have an experienced team, or at least a minimum number of experienced members, who can guide the design implementation.

These designers can direct critical design issues, such as clock handling, testability implementation, and efficient design techniques. Experienced linguistic personnel can also ensure the proper application of the HDL, verification language, and tools including synthesis, timing analysis, and layout.

A **design guideline** document that is easy to read, but defines the design rules can be very beneficial. This document addresses the coding standards, design styles, and synthesis design rules[v]. An incorrect design approach, such as use of multiple gated clocks, may prove a correct simulated model, but incorrect silicon hardware. Another timing example is the clock switching[vi] of asynchronous clocks.

A good and consistent **naming convention** helps in the minimization of errors because the name provides information about the objects and about the handling and timing information implied by those objects. Examples of information provided in the object naming include the class (e.g., constant, variable, signal, file, pointer), the polarity (e.g., active false) and the hardware significance (e.g., register or combinational).

For synthesizable designs, **exception handling** of error conditions may be detected and managed by the hardware with the possible cooperation of software. Exception handling conditions could represent conditions such as illegal operations (e.g., divide by zero, illegal instruction or command), or deadlock conditions (e.g., a longer than expected processing), or a subsystem hardware failure.

For testbench designs, exception handling of errors must be added in the code to address issues such as illegal access to files, errors in content of files, and errors in the emulation of environment surrounding the design under test.

Reuse was greatly discussed in books, articles, and vendor's brochures on what is or can be available for reuse, and tools that support reuse and integration (manual and automatic) of reused designs. Reuse is not limited to purchased components. Packages or designs[vii], verified as operational, can be placed in reuse libraries for use in multiple projects.

Tools support the design methodologies. Tools help in the automation for the creation of designs, and in the detection of errors failing to meet design rules. Classes of tools (considered MUST HAVE TOOLS) are listed in the following subsections. GNU tools, available at minimal distribution or no cost, are mentioned by name.

Language Sensitive Editors (LSEs) are must have tools because such editors understand the design language syntax, and can help in the mechanics (i.e., syntax) of the language. In addition, many LSEs provide automation in the fill-in of parameters (e.g., registers, sensitivity list), or the creation of templates for the definition of hierarchy and testbenches. *EMACS* is a GNU editor that operates on many platforms (PC Windows[viii], SUN OS, Linux, HP workstation, etc). It provides excellent features for many languages, including VHDL and Verilog. When *EMACS* is accompanied with *TSHELL (tcsh)*, it enables the user to operate in a *Unix*-like environment on any platform, thus maintaining compatibility and unification in the methods for compiling, searching, firing the simulators, and revision control (e.g., *RCS*).

For **design management**, version control software (such as GNU Revision Control System (RCS)) helps reduce errors because it automates the storing, retrieval, logging, identification, and merging of revisions. Having a log of what, when and why changes were made is very helpful in keeping track of changes, and for finding the "that used to work" errors. A configuration management system is necessary to maintain different versions of the "Intellectual Property" (IP) data[ix] (see reference for a discussion on that issue).

Linting is the process of identifying errors in coding rules, style, and design warnings. Several commercial linting tools are available. In addition, good synthesizer tools provide linting information, including synthesis-coding violations. The synthesizer provides information about the design including:

1. Identification of the registers
2. Unused inputs and outputs
3. Write-only hardware that gets optimized out
4. Graphical views of the design and interconnects for use as a sanity check of the hardware inferred by the HDL

Another class of tools that is very useful (when it is well implemented) is **the automatic generation of the design schematics** from the HDL[x] (RTL, behavioral, and testbench). The tool maintains the design hierarchy, and can generate selective schematics about the design. These schematics with selective flattening of the hierarchy, include the drive of selected signals up to their targets, and the tracing back (or sourcing) of information that drive a selected signal, or a section of design.

FSMs can be quite complex. There are commercial tools that can **extract the FSMs** from HDL code (at the RTL and behavioral levels), and represent these FSMs graphically. Some tools even go beyond that stage and identify errors in the FSMs, such as unreachable states, or trap states that cannot be exited without a hard reset.

Vendors are beginning to **support higher-level synthesis** with behavioral modeling **and automatic IP selection and insertion.**[xi] This technology enables a user to more quickly build and explore designs.

DESIGN VERIFICATION

A **transaction**[xii] is a specific sequence of states that an FSM traverses. Examples of transactions include READ operations, WRITE operations, and packet transmissions. A transaction has duration. In transaction based **stimulus generation** there is a modeling separation between the transactor (also called *client*[7]) that defines the sequence of transactions, and the *server* (known as Transaction Verification Model (*TVM*) in *Testbuilder*). With this capability, it is possible to construct large-scale tests that exercise *TVMs* or *servers* rather than supply all of the protocol details in a vector set. The advantages of this approach are:

- **Separation between the tasking** of jobs (e.g., WRITE) by the *client*, and the **execution** of the transactions (i.e., protocol, or twiddling of the many interface signals by *server*).
- **Reuse of client** when interface changes because of changes or testing of subblocks. The client is unchanged when the protocol changes.
- **Reuse of server** when different transactions or tasks are needed. The server is unchanged when the sequence of tasks (in client) is changed.

The application of **automatic verifier** models that verify compliance to requirements, and the reporting of design errors are absolute MUST HAVE in any design. **Coverage tools** provide an assessment of how well the design is exercised and help measure the progress and completeness of the verification effort. Coverage metrics include functional, branch, condition and expression, path, toggle, triggering, and signal-tracing coverages[xiii]. Many verification languages incorporate coverage to help measure the progress and completeness of the verification effort[9].

REVIEWS

All stages of the design process require thorough reviews by the design and application community to ensure design accuracy. These stages include the requirement specification, implementation plan, verification plan, detailed design (behavioral or RTL), synthesis, testbench, verification results, layout and timing, and documentation.

CONCLUSIONS AND RECOMMENDATIONS

Design errors can be contained with proper application of a process that puts heavy emphasis on the design planning and verification phases, and emphasizes strict design rules and review cycles. Use of proper tools is also essential to help in the design and verification automation. Support of experts in their respective fields (e.g., language, design, synthesis) short circuit errors early in the design phases.

REFERENCES

i Excerpted from *The American Heritage Dictionary of the English Language, Third Edition* Copyright © 1992 by Houghton Mifflin Company. Electronic version licensed from Lernout & Hauspie Speech Products N.V., further reproduction and distribution restricted in accordance with the Copyright Law of the United States. All rights reserved.

ii *Microsoft Press® Computer and Internet Dictionary* © & 1997, 1998 Microsoft Corporation. All rights reserved. Portions, *The Microsoft Press® Computer Dictionary, 3rd Edition*, Copyright © 1997 by Microsoft Press. All rights reserved.

iii For an example of the implementation of all the front-end design processes, see *Component Design by Example ... a Step-by-Step Process Using VHDL with UART as Vehicle*, Ben Cohen, 2001, ISBN 0-9705394-0-1.

iv *Writing testbenches, Functional verification of HDL models*, Janick Bergeron, Kluwer Academic Publishers 2000.

[v] *IEEE P1076.6 Standard For VHDL Register Transfer Level Synthesis* describes a standard syntax and semantics for VHDL RTL synthesis.

[vi] http://www.chip-guru.com/, article Switching Asynchronous Clocks, Vijay Nebhrajani.

[vii] The book *Component Design by Example*, Ben Cohen 2001, ISBN 0-9705394-0-1, provides a reusable VHDL TextIO command parsing package. The book *VHDL Coding Styles and Methodologies, 2nd Edition*, 1999 ISBN 0-7923-8474-1, provides reusable packages including a Linear Feedback Shift Register (LFSR) and a Conversion from various types to strings (Image).

[viii] EMACS with T-shell for PC is available on CD from http://www.opencores.org/OIPC/projects/OpenTech and in the book *Component Design by Example*, Ben Cohen 2001, ISBN 0-9705394-0-1.

[ix] See http://janick.bergeron.com/guild/1-01.html item 5 for suggestions for implementing IP repository.

[x] See Debussy tool at http://www.novas.com/ for an example of such a tool.

[xi] Also, see Y Explorations, Inc. web site http://www.yxi.com for more information on IP inference, insertion, and use HDL at higher level of abstractions.

[xii] http://www.testbuilder.net/

[xiii] *Verification Methodology Manual for Code Coverage in HDL Designs*, Michael Stuart & David Dempster, ISBN 0-9538-4820-5, Teamwork International 2000.